CERIAS

The Center for Education and Research in Information Assurance and Security

hexhive

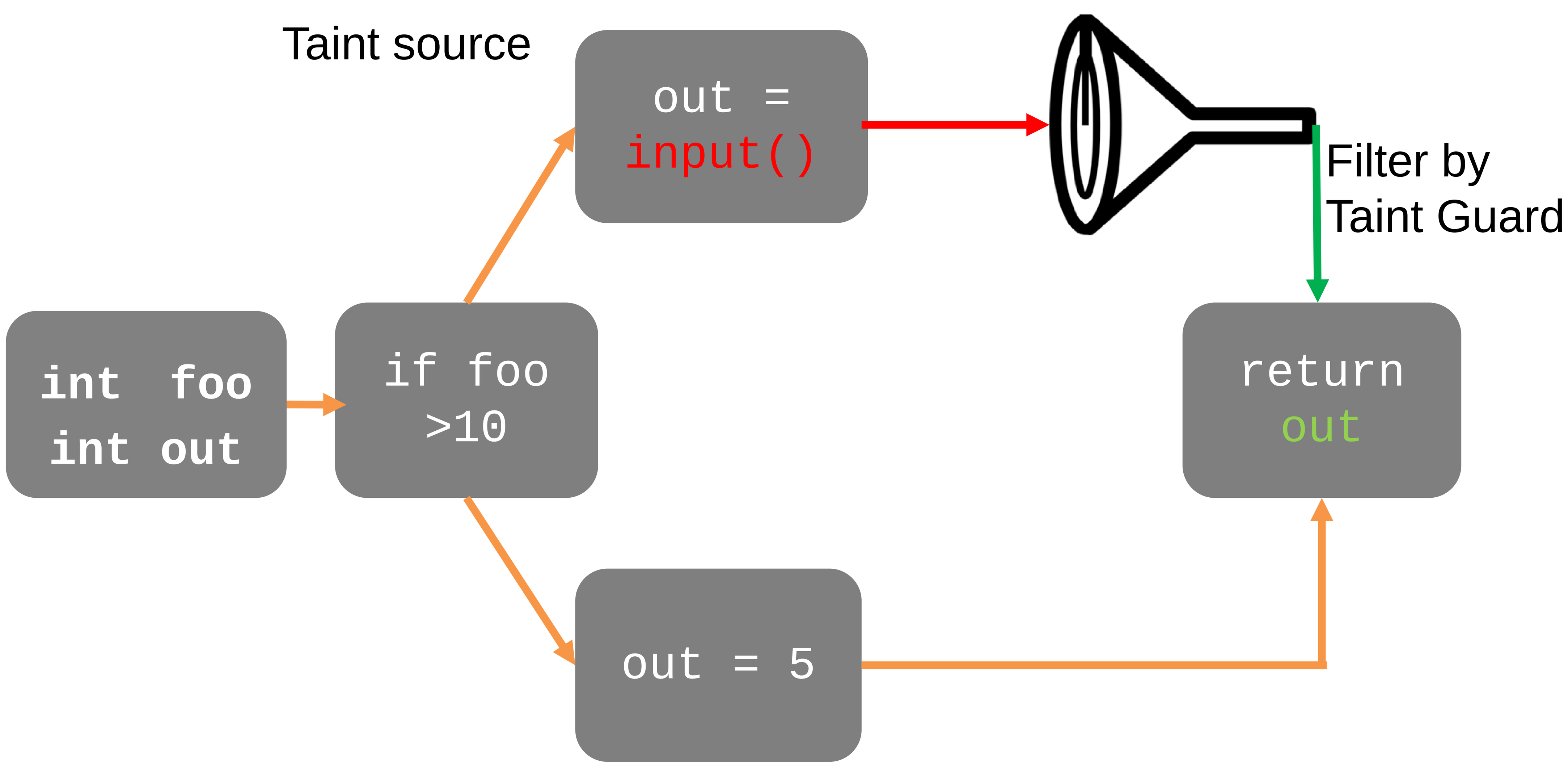
HexTaint: Ensuring Data Flow Integrity Using Dynamic Taint Analysis

Priyam Biswas Mathias Payer

Problem Statement

- The integrity and privacy of our data is threatened by security vulnerabilities in the programs that access the data
- Memory safety vulnerabilities such as buffer overflow attacks, use-after free attacks, and format string attacks accord for the majority of software vulnerabilities
- Again different logic errors and unanticipated data flows can also lead to data corruption
- Memory Safety vulnerabilities and Logic Errors allow an attacker to corrupt the data flow of a program and compromise the integrity and privacy of our data

Data Flow Path



Our Approach

Define operational semantics of the underlying taint-flow policy → Instrument the source program on LLVM bitcode level → Generate Data Flow Graph for taint tracking → Develop filters to safeguard data integrity

Dynamic Taint Analysis

A security tool used for monitoring the code during the run time and observing the effected code segments by previously determined taint sources

Challenges

- To generate appropriate filter
- To minimalize false positive
- To reduce overhead

Conclusion

- TaintGuard addresses data corruption to ensure data flow integrity
- Our implementation is in development phase, but it is expected to have low overhead

Highlights

- TaintGuard promises strong defense against data corruption
- More effective than traditional methods as the analysis is performed during run time
- LLVM Bitcode is an abstract bitstream container format as well as an encoding of LLVM IR (intermediate representation) into the container format

What is LLVM?

LLVM is a compiler infrastructure, written in C++, which is designed for compile-time, link-time, run-time, and "idle-time" optimization of programs written in arbitrary programming languages.

References

- Vijayakumar, Hayawardh, Xinyang Ge, Mathias Payer, and Trent Jaeger. "JIGSAW: Protecting resource access by inferring programmer expectations." In *Proceedings of the 23rd USENIX Security Symposium (Aug. 2014)*, pp. 973-988. 2014.

CERIAS

PURDUE UNIVERSITY